



# **ESP8266 SPI Overlap & Display Application Guide**

**Version 0.1**

Espressif Systems IOT Team

<http://bbs.espressif.com>

Copyright © 2015



### **Disclaimer and Copyright Notice**

Information in this document, including URL references, is subject to change without notice.

THIS DOCUMENT IS PROVIDED AS IS WITH NO WARRANTIES WHATSOEVER, INCLUDING ANY WARRANTY OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR ANY PARTICULAR PURPOSE, OR ANY WARRANTY OTHERWISE ARISING OUT OF ANY PROPOSAL, SPECIFICATION OR SAMPLE. All liability, including liability for infringement of any proprietary rights, relating to use of information in this document is disclaimed. No licenses express or implied, by estoppel or otherwise, to any intellectual property rights are granted herein.

The WiFi Alliance Member Logo is a trademark of the WiFi Alliance.

All trade names, trademarks and registered trademarks mentioned in this document are property of their respective owners, and are hereby acknowledged.

**Copyright © 2015 Espressif Systems. All rights reserved.**



# Table of Contents

<b>1. Preambles .....</b>	<b>4</b>
<b>2. Hardware Connection of SPI Overlap Mode .....</b>	<b>5</b>
<b>3. API Description of SPI Overlap Mode .....</b>	<b>5</b>
<b>4. Display Screen Console Program DEMO .....</b>	<b>6</b>
4.1. Connection Description .....	6
4.2. API Function Description.....	6
4.3. Pre-compiled Macro Setting .....	8



## 1. Preambles

The Overlap mode of ESP8266 Host SPI allows for two SPI modes (SPI and HSPI) to reuse the same IO interface (such as SCLK, MOSI and MISO) for the operation of multiple slave SPI devices. The hardware supports 3 line chip selection. If there are additional 3 slave devices, GPIO can be adopted as CS signal for the communication of multiple slave device.

Generally speaking, in order to ensure that the CPU can be running at high efficiency, SPI module is used to read the running program from external Flash to CPU CACHE, while HSPI module is used to operate slave devices of other users. Under Overlap mode, the hardware will automatically arbitrate the control of two SPI modules to the current pin signal for time-sharing application. If the software starts HSPI communication, the arbitration signal will delay the start of HSPI block communication via the working of SPI. The arbitration signal is then allowed to start the communication of HSPI IO interface after SPI finishes reading the program codes for communication. This is illustrated in Figure 1. For user software, only a switch of the corresponding CS signal before the start of communicator is needed. Other operations are of no difference to the use of single HSPI communication.

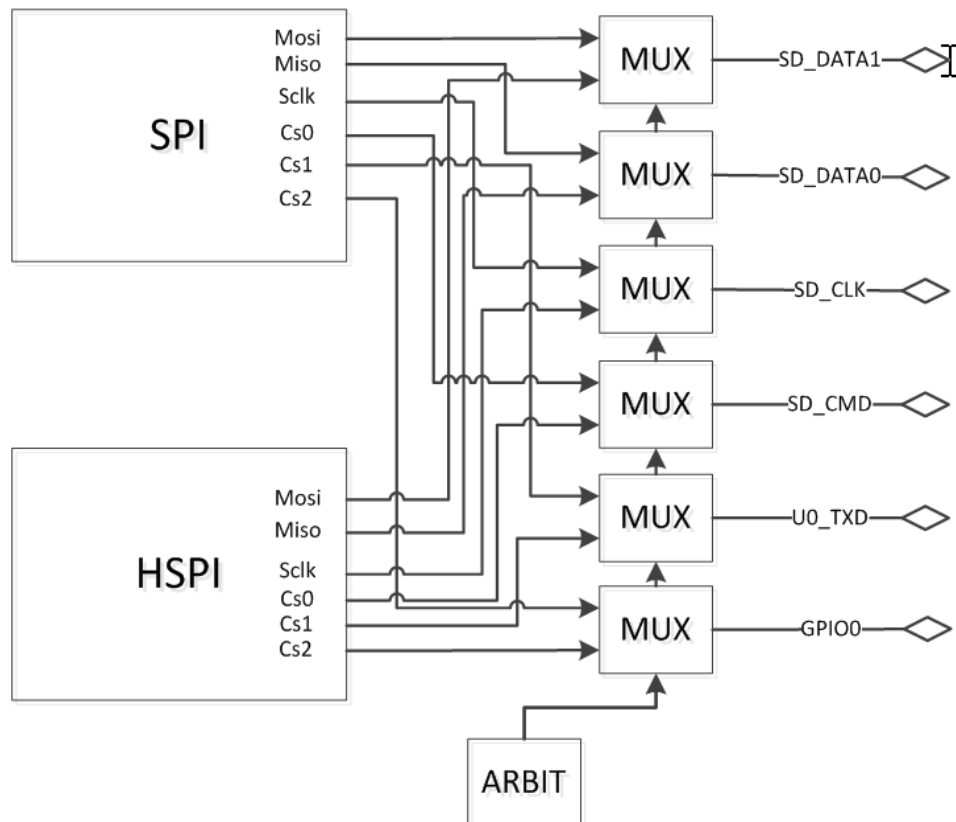


Figure 1 SPI Overlap Block Diagram

Please refer to EPS8266 SPI Module User Guide for more information about the application method of Host SPI Module. The configuration method of Overlap mode is discussed in detail below.



## 2. Hardware Connection of SPI Overlap Mode

Pins including SD\_CLK, SD\_DATA0, and SD\_DATA1 correspond to pins SCLK, MISO and MOSI in two SPI modes, while pins SD\_CMD, U0TXD, and GPIO0 correspond to chip selection (CS) signals CS0, CS1, and CS2 respectively. Generally, SD\_CMD connects to the CS signal of an external Flash, while U0TXD and GPIO0 can be connect with the CS signals of two slave devices. It can connect to the CS signal of two slave devices. Besides, HSPI can read and write Flash data through enabled CS0, independent of SPI (e.g. Read some pre-stored user data).

If more SPI devices are needed, device can be selected via other GPIOs, while CS0, CS1, and CS2 are blocked by the configuration register.

## 3. API Description of SPI Overlap Mode

```
void hapi_Overlap_init(void)
```

**Function:** After SPI Overlap mode has been initialized, and SPI and HSPI interfaces are invoked, interfaces including CLK, MOSI, and MISO can be shared with SPI and HSPI interfaces to communicate with different devices. By default, CS2 is the CS signal of HSPI interface. Please be careful when switching CS signals during communication.

**Location:** `\app\user\user_main.c` in the DEMO.

```
SELECT_OLED(), SELECT_TFT()
```

**Function:** Switch the CS pin of HSPI and OLED in DEMO connects to CS2. TFTLCD connects to CS1. Before the start of HSPI communication, macro needs to be called. The macro definition is as follows:

```
#define SELECT_OLED()  CLEAR_PERI_REG_MASK(SPI_PIN(HSPI), SPI_CS2_DIS);\
                      SET_PERI_REG_MASK(SPI_PIN(HSPI), SPI_CS0_DIS |SPI_CS1_DIS)
#define SELECT_TFT()  CLEAR_PERI_REG_MASK(SPI_PIN(HSPI), SPI_CS1_DIS);\
                      SET_PERI_REG_MASK(SPI_PIN(HSPI), SPI_CS0_DIS |SPI_CS2_DIS)
```

Therefore, users can change the macro definition. For example, the following macro can be defined if HSPI is used to the operate Flash:

```
#define SELECT_FLASH()  CLEAR_PERI_REG_MASK(SPI_PIN(HSPI), SPI_CS0_DIS);\
                      SET_PERI_REG_MASK(SPI_PIN(HSPI), SPI_CS1_DIS |SPI_CS2_DIS)
```

If normal GPIO is used for CS, the following is needed:

```
#define DISABLE_CS()\
                      SET_PERI_REG_MASK(SPI_PIN(HSPI), SPI_CS0_DIS |SPI_CS1_DIS |SPI_CS2_DIS)
```



**Location:** `\app\include\user_lcd.h` in the DEMO.

Please refer to *ESP8266 Host SPI User Guide* for more information about other host SPI communication.

## 4. Display Screen Console Program DEMO

The DEMO is used to print simple strings on display screens, including LCD for parameter display and debug printing. DEMO driver supports two screens currently, i.e. 3.5-inch TM035PDZV36 480\*320 TFT colored LCD and Zhong JY. Tech 1.3-inch 128\*64OLED. The driver programs can communicate with the display screen via ESP8266 HSPI interface under Overlap mode.

Under SPI Overlap mode, the two screens and 8266 external program flash chip share SCLK, MOSI and MISO signals on the SPI bus. Different CS signals are used in different device.

### 4.1. Connection Description

#### Zhong JY. Tech 1.3-inch OLED Connection

The signals in OLED, i.e. SCLK, MOSI, CS, DC, RESET connects to the pins in 8266, i.e. SD\_CLK, SD\_DATA1, GPIO0, MTCK, GPIO5 respectively. The VCC in OLED and GND connects to 3.3V network and GND on DEMO board.

#### Tian Ma 3.5-inch TFT LCD

The signals in TFT, i.e. SCLK, MOSI, CS, RESET connects to the pins in 8266, i.e. SD\_CLK, SD\_DATA1, U0TXD, GPIO5 respectively. The VCC in OLED and GND connects to 3.3V network and GND on DEMO board.

### 4.2. API Function Description

```
void screen_init(void)
```

**Function:** Display screen initialization program. Call the function after it is enabled.

**Location:** `\app\user\user_lcd.c` and `\app\include\user_lcd.h`

```
void scr_param_config(uint8 bkg_color,uint8 ft_color,uint8 ft_size, uint8  
scr_size_clr_row, uint8 scr_size_x,uint8 scr_size_y)
```

**Function:** display parameter for the global variable configuration string of the `scr_font_param` structure.

**Parameter description:**



Parameter	Description
uint8 bkg_color	Background color of TFT can change between BLACK_8COLOR and WHITE_8COLOR. Do not use OLED display screen.
uint8 ft_color	Font color of TFT can change between BLACK_8COLOR and WHITE_8COLOR. Do not use OLED display screen.
uint8 ft_size	Font size with 12*6 ASCII character. The parameter is the multiple of pixels under the character. For example, if ft_size is 2, the actual font size is 24*12. Input non-zero value.
uint8 scr_size_clr_row	Rows should be removed after the screen is refreshed. Input non-zero value.
uint8 scr_size_x	Each line shows the character number. Please note that it should not exceed the pixel range of the screen.
uint8 scr_size_y	This parameter shows the character lines. Please note that it should not exceed the pixel range of the screen.

**Location:** `\app\user\user_lcd.c` and `\app\include\user_lcd.h`, call in the function `screen_init`

```
void scr_printf(const char* fmt, ...)
```

**Function:** used for standard printing of functions displayed on the screen, similar to the using method of `printf` in C programming language

**Parameter description:**

`const char* fmt`: shows the character string.

`...`: variable parameters that needs to be displayed in the corresponding string

**Location:** `\app\user\user_lcd.c` and `\app\include\user_lcd.h`

```
void at_lcd_print(uint8* str)
```

**Function:** shows the assigned character string displayed on the screen order.

`uint8* str`: the starting address of string array.



### 4.3. Pre-compiled Macro Setting

```
#define OLED_SCR      1
#define TFT_SCR       1
#define Overlap_TEST  0
```

**Location:** [\app\include\user\\_lcd.h](#)

OLED\_SCR and TFT\_SCR can control the debugging characters displayed on the corresponding screen. The program supports the same character shown in two screens. Overlap\_TEST is used for SPI Overlap test when TFT is used to display image. TFT should be set at 0 as it conflicts with the displayed characters.